EXHIBIT 1

bytemaster / **EOSIO-Documentation**

forked from wanderingbort/EOSIO-Documentation

EOS.IO Technical White Paper

DRAFT: June 5, 2017

Abstract: The EOS.IO software introduces a new blockchain architecture designed to enable vertical and horizontal scaling of decentralized applications. This is achieved by creating an operating system-like construct upon which applications can be built. The software provides accounts, authentication, databases, asynchronous communication and the scheduling of applications across hundreds of CPU cores or clusters. The resulting technology is a blockchain architecture that scales to millions of transactions per second, eliminates user fees, and allows for quick and easy deployment of decentralized applications.

Copyright © 2017 block.one

Without permission, anyone may use, reproduce or distribute any material in this whitepaper for non-commercial and educational use (i.e., other than for a fee or for commercial purposes) provided that the original source and the applicable copyright notice are cited.

DISCLAIMER: This draft EOS.IO Technical Whitepaper is for information purposes only. block.one does not guarantee the accuracy of the conclusions reached in this paper, and the whitepaper is provided “as is” with no representations and warranties, express or implied, whatsoever, including, but not limited to: (i) warranties of merchantability, fitness for a particular purpose, title or noninfringement; (ii) that the contents of this whitepaper are free from error or suitable for any purpose; and (iii) that such contents will not infringe third-party rights. All warranties are expressly disclaimed. block.one and its affiliates expressly disclaim all liability for and damages of any kind arising out of the use, reference to, or reliance on any information contained in this whitepaper, even if advised of the possibility of such damages. In no event will block.one or its affiliates be liable to any person or entity for any direct, indirect, special or consequential damages for the use of, reference to, or reliance on this whitepaper or any of the content contained herein.

Background

Blockchain technology was introduced in 2008 with the launch of the bitcoin currency, and since then entrepreneurs and developers have been attempting to generalize the technology in order to support a wider range of applications on a single blockchain platform.

While a number of blockchain platforms have struggled to support functional decentralized applications, application specific blockchains such as the BitShares decentralized exchange (2014) and Steem social media platform (2016) have become heavily used blockchains with tens of thousands of daily active users. They have achieved this by increasing performance to thousands of transactions per second, reducing latency to 1.5 seconds, eliminating fees, and providing a user experience similar to those currently provided by existing centralized services.

Existing blockchain platforms are burdened by large fees and limited computational capacity that prevent widespread blockchain adoption.

Requirements for Blockchain Applications

In order to gain widespread use, applications on the blockchain require a platform that is flexible enough to meet the following requirements:

Support Millions of Users

Disrupting businesses such as Ebay, Uber, AirBnB, and Facebook, require blockchain technology capable of handling tens of millions of active daily users. In certain cases, applications may not work unless a critical mass of users is reached and therefore a platform that can handle mass number of users is paramount.

Free Usage

Application developers need the flexibility to offer users free services; Users should not have to pay in order to use the platform or benefit from its services. A blockchain platform that is free to use for users will likely gain more widespread adoption. Developers and businesses can then create effective monetization strategies.

Easy Upgrades and Bug Recovery

Businesses building blockchain based applications need the flexibility to enhance their applications with new features.

All non-trivial software is subject to bugs, even with the most rigorous of formal verification. The platform must be robust enough to fix bugs when they inevitably occur.

Low Latency

A good user experience demands reliable feedback with delay of no more than a few seconds. Longer delays frustrate users and make applications built on a blockchain less competitive with existing non-blockchain alternatives.

Sequential Performance

There are some applications that just cannot be implemented with parallel algorithms due to sequentially dependent steps. Applications such as exchanges need enough sequential performance to handle high volumes and therefore a platform with fast sequential performance is required.

Parallel Performance

Large scale applications need to divide the workload across multiple CPUs and computers.

Consensus Algorithm (DPOS)

EOS.IO software utilizes the only decentralized consensus algorithm capable of meeting the performance requirements of applications on the blockchain, Delegated Proof of Stake (DPOS). Under this algorithm, those who hold tokens on a blockchain may select block producers through a continuous approval voting system and anyone may choose to participate in block production and will be given an opportunity to produce blocks proportional to the total votes they have received relative to all other producers. For private blockchains the management will use the tokens to add and remove IT staff.

The EOS.IO software enables blocks to be produced exactly every 3 seconds and exactly one producer is authorized to produce a block at any given point in time. If the block is not produced at the scheduled time then the block for that time slot is skipped. When one or more blocks are skipped, there is a 6 or more second gap in the blockchain.

Using the EOS.IO software blocks are produced in rounds of 21. At the start of each round 21 unique block producers are chosen. The top 20 by total approval are automatically chosen every round and the last producer is chosen proportional to their number of votes relative to other producers. The selected producers are shuffled using a pseudorandom number derived from the block time. This shuffling is done to ensure that all producers maintain balanced connectivity to all other producers.

If a producer misses a block and has not produced any block within the last 24 hours they are removed from consideration until they notify the blockchain of their intention to start producing blocks again. This ensures the network operates smoothly by minimizing the number of blocks missed by not scheduling those who are proven to be unreliable.

Under normal conditions a DPOS blockchain does not experience any forks because the block producers cooperate to produce blocks rather than compete. In the event there is a fork, consensus will automatically switch to the longest chain. This metric works because the rate at which blocks are added to a blockchain chain fork is directly correlated to the percentage of block producers that share the same consensus. In other words, a blockchain fork with more producers on it will grow in length faster than one with fewer producers. Furthermore, no block producer should be producing blocks on two forks at the same time. If a block producer is caught doing this then such block producer will likely be voted out. Cryptographic evidence of such double-production may also be used to automatically remove abusers.

Transaction Confirmation

Typical DPOS blockchains have 100% block producer participation. A transaction can be considered confirmed with 99.9% certainty after an average of 1.5 seconds from time of broadcast.

There are some extraordinary cases where a software bug, Internet congestion, or a malicious block producer will create two or more forks. For absolute certainty that a transaction is irreversible, a node may choose to wait for confirmation by 15 out of the 21 block producers. Based on a typical configuration of the EOS.IO software, this will take an average of 45 seconds under normal circumstances. By default all nodes will consider a block confirmed by 15 of 21 producers irreversible and will not switch to a fork that excludes such a block regardless of length.

It is possible for a node to warn users that there is a high probability that they are on a minority fork within 9 seconds of the start of a fork. After 2 consecutive missed blocks there is a 95% probability a node is on a minority fork. With 3 consecutive missed blocks there is a 99% certainty of being on a minority fork. It is possible to generate a robust predictive model that will utilize information about which nodes missed, recent participation rates, and other factors to quickly warn operators that something is wrong.

The response to such a warning depends entirely upon the nature of the business transactions, but the simplest response is to wait for 15/21 confirmations until the warning stops.

Transaction as Proof of Stake (TaPoS)

The EOS.IO software requires every transaction to include the hash of a recent block header. This hash serves two purposes:

1. prevents a replay of a transaction on forks that do not include the referenced block; and
2. signals the network that a particular user and their stake are on a specific fork.

Over time all users end up directly confirming the blockchain which makes it difficult to forge counterfeit chains as the counterfeit would not be able to migrate transactions from the legitimate chain.

Accounts

The EOS.IO software permits all accounts to be referenced by a unique human readable name of 2 to 32 characters in length. The name is chosen by the creator of the account. All accounts must be funded with the minimal account balance at the time they are created to cover the cost of storing account data. Account names also support namespaces such that the owner of account @domain is the only one who can create the account @user.domain.

In a decentralized context, application developers will pay the nominal cost of account creation to sign up a new user. Traditional businesses already spend significant sums of money per customer they acquire in the form of advertizing, free services, etc. The cost of funding a new blockchain account should be insignificant in comparison. Fortunately, there is no need to create accounts for users already signed up by another application.

Messages & Handlers

Each account can send structured messages to other accounts and may define scripts to handle messages when they are received. The EOS.IO software gives each account its own private database which can only be accessed by its own message handlers. Message handling scripts can also send messages to other accounts. The combination of messages and automated message handlers is how EOS.IO defines smart contracts.

Role Based Permission Management

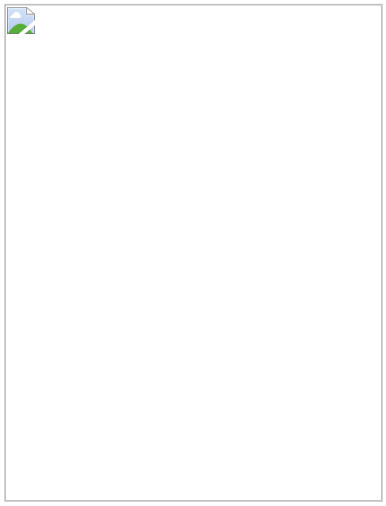
Permission management involves determining whether or not a message is properly authorized. The simplest form of permission management is checking that a transaction has the required signatures, but this implies that required signatures are already known. Generally authority is bound to individuals or groups of individuals and is often compartmentalized. The EOS.IO software provides a declarative permission management system that gives accounts fine grained and high level control over who can do what and when.

It is critical that authentication and permission management be standardized and separate from the business logic of the application. This enables tools to be developed to manage permissions in a general purpose manner and also provide significant opportunities for performance optimization.

Every account may be controlled by any weighted combination of other accounts and private keys. This creates a hierarchical authority structure that reflects how permissions are organized in reality, and makes multi-user control over funds easier than ever. Multi-user control is the single biggest contributor to security, and, when used properly, it can greatly eliminate the risk of theft due to hacking.

EOS.IO software allows accounts can define what combination of keys and/or other accounts can send a particular message type to another account. For example, it is possible to have one key for a user's social media account and another for access to the exchange. It is even possible to give other accounts permission to act on behalf of a user's account without assigning them keys.

Named Permission Levels



Using the EOS.IO software, accounts can define named permission levels each of which can be derived from higher level named permissions. Each named permission level defines an authority; an authority is a threshold multi-signature check consisting of keys and/or named permission levels of other accounts. For example, an account's "Friend" permission level can be set for the account to be controlled equally by any of the account's friends.

Another example is the Steem blockchain which has three hard-coded named permission levels: owner, active, and posting. The posting permission can only perform social actions such as voting and posting, while the active permission can do everything except change the owner. The owner permission is meant for cold storage and is able to do everything. EOS.IO generalizes this concept by allowing each account

holder to define their own hierarchy as well as the grouping of actions.

Named Message Handler Groups

The EOS.IO software allows each account to organize its own message handlers into named and nested groups. These named message handler groups can be referenced by other accounts when they configure their permission levels.

The highest level message handler group is the account name and the lowest level is the individual message type being received by the account. These groups can be referenced like so: **@accountname.groupa.subgroupb.MessageType**.

Under this model it is possible for an exchange contract to group order creation and canceling separately from deposit and withdraw. This grouping by the exchange contract is a convenience for users of the exchange.

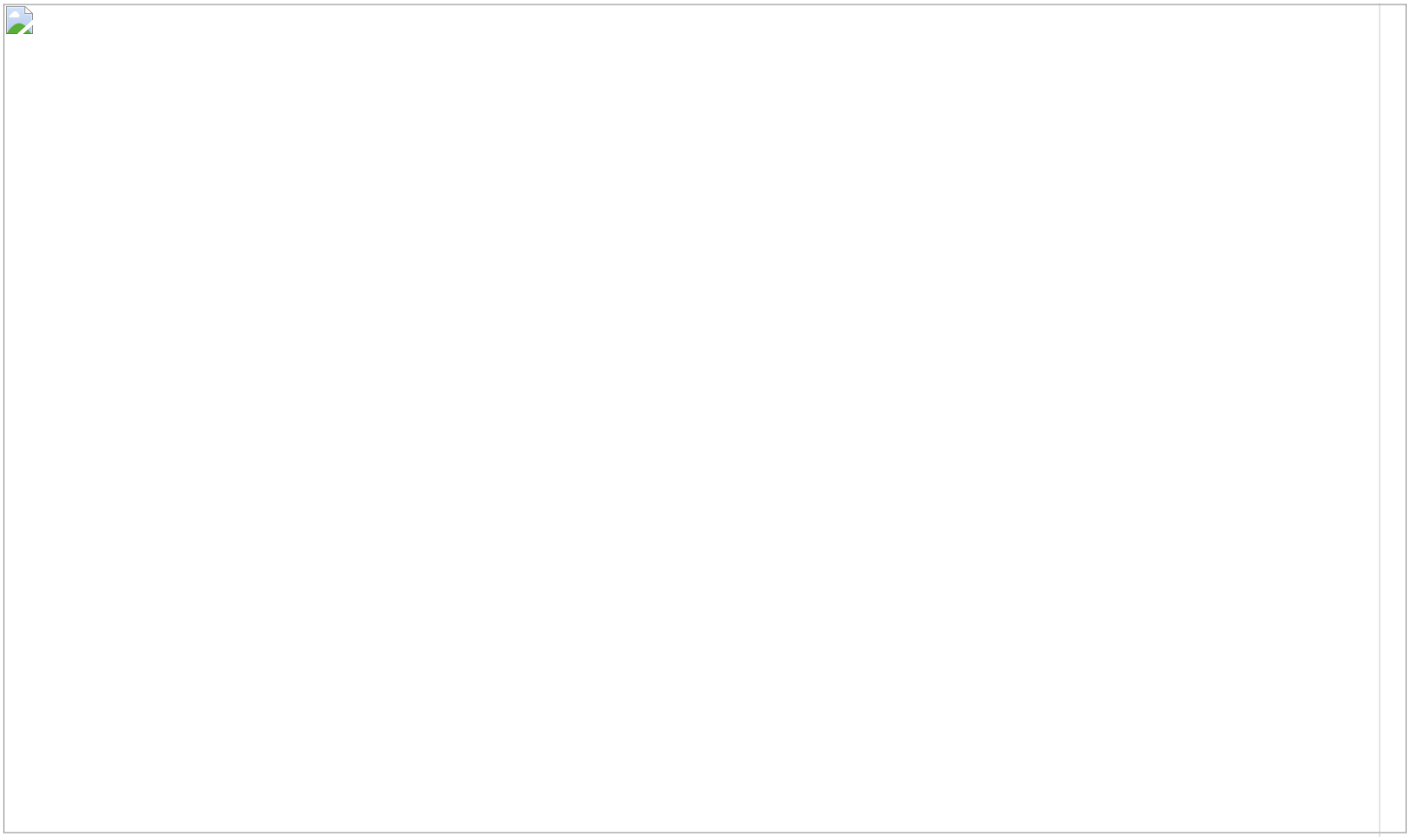
Permission Mapping

EOS.IO software allows each account to define a mapping between a Named Message Handler Group of any account and their own Named Permission Level. For example, an account holder could map the account holder's social media application to the account holder's "Friend" permission group. With this mapping, any friend could post as the account holder on the account holder's social media. Even though they would post as the account holder, they would still use their own keys to sign the message. This means it is always possible to identify which friends used the account and in what way.

Evaluating Permissions

When delivering a message of type "**Action**", from **@alice** to **@bob** the EOS.IO software will first check to see if **@alice** has defined a permission mapping for **@bob.groupa.subgroup.Action**. If nothing is found then a mapping for **@bob.groupa.subgroup** then **@bob.groupa**, and lastly **@bob** will be checked. If no further match is found, then the assumed mapping will be to the named permission group **@alice.active**.

Once a mapping is identified then signing authority is validated using the threshold multi-signature process and the authority associated with the named permission. If that fails, then it traverses up to the parent permission and ultimately to the owner permission, **@alice.owner**.



Default Permission Groups

The technology also allows all accounts have an "owner" group which can do everything, and an "active" group which can do everything except change the owner group. All other permission groups are derived from "active".

Parallel Evaluation of Permissions

The permission evaluation process is "read-only" and changes to permissions made by transactions do not take effect until the end of a block. This means that all keys and permission evaluation for all transactions can be executed in parallel. Furthermore, this means that a rapid validation of permission is possible without starting the costly application logic that would have to be rolled back. Lastly, it means that transaction permissions can be evaluated as pending transactions are received and do not need to be re-evaluated as they are applied.

All things considered, permission verification represents a significant percentage of the computation required to validate transactions. Making this a read-only and trivially parallelizable process enables a dramatic increase in performance.

When replaying the blockchain to regenerate the deterministic state from the log of messages there is no need to evaluate the permissions again. The fact that a transaction is included in a known good block is sufficient to skip this step. This dramatically reduces the computational load associated with replaying an ever growing blockchain.

Messages with Mandatory Delay

Time is a critical component of security. In most cases, it is not possible to know if a private key has been stolen until it has been used. Time based security is even more critical when people have applications that require keys be kept on computers connected to the internet for daily use. The EOS.IO software enables application developers to indicate that certain messages must wait a minimum period of time after being included in a block before they can be applied. During this time they can be canceled.

Users can then receive notice via email or text message when one of these messages is broadcast. If they did not authorize it, then they can use the account recovery process to recover their account and retract the message.

The required delay depends upon how sensitive an operation is. Paying for a coffee can have no delay and be irreversible in seconds, while buying a house may require a 72 hour clearing period. Transferring an entire account to new control may take up to 30 days. The exact delays chosen are up to application developers and users.

Recovery from Stolen Keys

The EOS.IO software provides users a way to restore control of their account when their keys are stolen. An account owner can use any owner key that was active in the last 30 days along with approval from their designated account recovery partner to reset the owner key on their account. The account recovery partner cannot reset control of the account without the help of the owner.

There is nothing for the hacker to gain by attempting to go through the recovery process because they already "control" the account. Furthermore, if they did go through the process, the recovery partner would likely demand identification and multi-factor authentication (phone and email). This would likely compromise the hacker or gain the hacker nothing in the process.

This process is also very different from a simple multi-signature arrangement. With a multi-signature transaction, there is another company that is party to every transaction that is executed, but with the recovery process the agent is only a party to the recovery process and has no power over the day-to-day transactions. This dramatically reduces costs and legal liabilities for everyone involved.

Deterministic Parallel Execution of Applications

Blockchain consensus depends upon deterministic (reproducible) behavior. This means all parallel execution must be free from the use of mutexes or other locking primitives. Without locks there must be some way to guarantee that all accounts can only read and write their own private database. It also means that each account processes messages sequentially and that parallelism will be at the account level.

Using the EOS.IO software, it is the job of the block producer to organize message delivery into independent threads so that they can be evaluated in parallel. The state of each account depends only upon the messages delivered to it. The schedule is the output of a block producer and will be deterministically executed, but the process for generating the schedule need not be deterministic. This means that block producers can utilize parallel algorithms to schedule transactions.

Part of parallel execution means that when a script generates a new message it does not get delivered immediately, instead it is scheduled to be delivered in the next cycle. The reason it cannot be delivered immediately is because the receiver may be actively modifying its own state in another thread.

Minimizing Communication Latency

Latency is the time it takes for one account to send a message to another account and then receive a response. The goal is to enable two accounts to exchange messages back and forth within a single block without having to wait 3 seconds between each message. To enable this, the EOS.IO software divides each block into cycles. Each cycle is divided into threads and each thread contains a list of transactions. Each transaction contains a set of messages to be delivered. This structure can be visualized as a tree where alternating layers are processed sequentially and in parallel.

```
Block
  Cycles (sequential)
    Threads (parallel)
      Transactions (sequential)
        Messages (sequential)
          Receiver and Notified Accounts (parallel)
```

Transactions generated in one cycle can be delivered in any subsequent cycle or block. Block producers will keep adding cycles to a block until the maximum wall clock time has passed or there are no new generated transactions to deliver.

It is possible to use static analysis of a block to verify that within a given cycle no two threads contain transactions that modify the same account. So long as that invariant is maintained a block can be processed by running all threads in parallel.

Read-Only Message Handlers

Some accounts may be able to process a message on a pass/fail basis without modifying their internal state. If this is the case then these handlers can be executed in parallel so long as only read-only message handlers for a particular account are included in one or more threads within a particular cycle.

Atomic Transactions with Multiple Accounts

Sometimes it is desirable to ensure that messages are delivered to and accepted by multiple accounts atomically. In this case both messages are placed in one transaction and both accounts will be assigned the same thread and the messages applied sequentially. This situation is not ideal for performance and when it comes to "billing" users for usage, they will get billed by the number of unique accounts referenced by a transaction.

For performance and cost reasons it is best to minimize atomic operations involving two or more heavily utilized accounts.

Partial Evaluation of Blockchain State

Scaling blockchain technology necessitates that components are modular. Everyone should not have to run everything, especially if they only need to use a small subset of the applications.

An exchange application developer runs full nodes for the purpose of displaying the exchange state to its users. This exchange application has no need for the state associated with social media applications. EOS.IO software allows any full node to pick any subset of applications to run. Messages delivered to other applications are safely ignored because an application's state is derived entirely from the messages that are delivered to it.

This has some significant implications on communication with other accounts. Most significantly it cannot be assumed that the state of the other account is accessible on the same machine. It also means that while it is tempting to enable "locks" that allow one account to synchronously call another account, this design pattern breaks down if the other account is not resident in memory.

All state communication among accounts must be passed via messages included in the blockchain.

Subjective Best Effort Scheduling

The EOS.IO software cannot obligate block producers to deliver any message to any other account. Each block producer makes their own subjective measurement of the computational complexity and time required to process a transaction. This applies whether a transaction is generated by a user or automatically by a script.

The EOS.IO software provides that at a network level all transactions are billed a fixed computational bandwidth cost regardless of whether it took .01ms or a full 10 ms to execute it. However, each individual block producer using the software may calculate resource usage using their own algorithm and measurements. When a block producer concludes that a transaction or account has consumed a disproportionate amount of the computational capacity they simply reject the transaction when producing their own block; however, they will still process the transaction if other block producers consider it valid.

In general so long as even 1 block producer considers a transaction as valid and under the resource usage limits then all other block producers will also accept it, but it may take up to 1 minute for the transaction to find that producer.

In some cases a producer may create a block that includes transactions that are an order of magnitude outside of acceptable ranges. In this case the next block producer may opt to reject the block and the tie will be broken by the third producer. This is no different than what would happen if a large block caused network propagation delays. The community would notice a pattern of abuse and eventually remove votes from the rogue producer.

This subjective evaluation of computational cost frees the blockchain from having to precisely and deterministically measure how long something takes to run. With this design there is no need to precisely count instructions which dramatically increases opportunities for optimization without breaking consensus.

Token Model and Resource Usage

All blockchains are resource constrained and require a system to prevent abuse. With the EOS.IO software, there are three broad classes of resources that are consumed by applications:

1. Bandwidth and Log Storage (Disk);
2. Computation and Computational Backlog (CPU); and
3. State Storage (RAM).

Bandwidth and computation have two components, instantaneous usage and long-term usage. A blockchain maintains a log of all messages and this log is ultimately stored and downloaded by all full nodes. With the log of messages it is possible to reconstruct the state of all applications.

The computational debt is calculations that must be performed to regenerate state from the message log. If the computational debt grows too large then it becomes necessary to take snapshots of the blockchain's state and discard the blockchain's history. If computational debt grows too quickly then it may take 6 months to replay 1 year worth of transactions. It is critical, therefore, that the computational debt be carefully managed.

Blockchain state storage is information that is accessible from application logic. It includes information such as order books and account balances. If the state is never read by the application then it should not be stored. For example, blog post content and comments are not read by application logic so they should not be stored in the blockchain's state. Meanwhile the existence of a post/comment, the number of votes, and other properties do get stored as part of the blockchain's state.

Block producers publish their available capacity for bandwidth, computation, and state. The EOS.IO software allows each account to consume a percentage of the available capacity proportional to the amount of tokens held in a 3-day staking contract. For example, if a blockchain based on the EOS.IO software is launched and if an account holds 1% of the total tokens distributable pursuant to that blockchain, then that account has the potential to utilize 1% of the state storage capacity.

Using the EOS.IO software, bandwidth and computational capacity are allocated on a fractional reserve basis because they are transient (unused capacity cannot be saved for future use). The algorithm used by EOS.IO is similar to the algorithm used by Steem to rate-limit bandwidth usage.

Objective and Subjective Measurements

As discussed earlier, instrumenting computational usage has a significant impact on performance and optimization; therefore, all resource usage constraints are ultimately subjective and enforcement is done by block producers according to their individual algorithms and estimates.

That said, there are certain things that are trivial to measure objectively. The number of messages delivered and the size of the data stored in the internal database are cheap to measure objectively. The EOS.IO software enables block producers to apply the same algorithm over these objective measures but may choose to apply stricter subjective algorithms over subjective measurements.

Receiver Pays

Traditionally, it is the business that pays for office space, computational power, and other costs required to run the business. The customer buys specific products from the business and the revenue from those product sales is used to cover the business costs of operation. Similarly, no website obligates its visitors to make micropayments for visiting its website to cover hosting costs. Therefore, decentralized applications should not force its customers to pay the blockchain directly for the use of the blockchain.

The EOS.IO software does not require its users to pay directly to the blockchain for its use and therefore does not constrain or prevent t a business from determining its own monetization strategy for its products.

Delegating Capacity

If a blockchain is launched using the EOS.IO software and tokens are held by a holder who may not have an immediate need to consume all or part of the available bandwidth, such holder can choose to give or rent the unconsumed bandwidth to others; the block producers running EOS.IO software will recognize this delegation of capacity and allocate bandwidth accordingly.

Separating Transaction costs from Token Value

One of the major benefits of the EOS.IO software is that the amount of bandwidth available to an application is entirely independent of any token price. If an application owner holds a relevant number of tokens, then the application can run indefinitely within a fixed state and bandwidth usage. Developers and users are unaffected from any price volatility in the token market and therefore not reliant on a price feed. The EOS.IO software enables block producers to naturally increase bandwidth, computation, and storage available per token independent of the token's value.

The EOS.IO software awards block producers tokens every time they produce the block. The value of the tokens will impact the amount of bandwidth, storage, and computation a producer can afford to purchase; this model naturally leverages rising token values to increase network performance.

State Storage Costs

While bandwidth and computation can be delegated, storage of application state will require an application developer to hold tokens until that state is deleted. If state is never deleted then the tokens are effectively removed from circulation.

Every user account requires a certain amount of storage; therefore, every account must maintain a minimum balance. As storage capacity of the network increases this minimum required balance will fall.

Block Rewards

EOS.IO software awards new tokens to a block producer every time a block is produced. The number of tokens created is determined by the median of the desired pay published by all block producers. The EOS.IO software may be configured to enforce a cap on producer awards such that the total annual increase in token supply does not exceed 5%.

Community Benefit Applications

In addition to electing block producers, based on the EOS.IO software, users can elect 3 community benefit applications also known as smart contracts. These 3 applications will receive tokens of up to a configured percent of the token supply per annum minus the tokens that have been paid to block producers. These smart contracts will receive tokens proportional to the votes each application has received from token holders. The elected applications or smart contracts can be replaced by newly elected applications or smart contracts by token holders.

Governance

Governance is the process by which people reach consensus on subjective matters that cannot be captured entirely by software algorithms. The EOS.IO software implements a governance process that efficiently directs the existing influence of block producers. Absent a defined governance process, prior blockchains relied ad hoc, informal, and often controversial governance processes that result in unpredictable outcomes.

The EOS.IO software recognizes that power originates with the token holders who delegate that power to the block producers. The block producers are given limited and checked authority to freeze accounts, update defective applications, and propose hard forking changes to the underlying protocol.

Part of the EOS.IO software is the election of block producers. Before any change can be made to the blockchain these block producers must approve it. If the block producers refuse to make changes desired by the token holders then they can be voted out. If the block producers make changes without permission of the token holders then all other non-producing full-node validators (exchanges, etc) will reject the change.

Freezing Accounts

Sometimes a smart contact behaves in an aberrant or unpredictable manner and fails to perform as intended; other times an application or account may discover an exploit that enables it to consume an unreasonable amount of resources. When such issues inevitably occur, the block producers have the power to rectify such situations.

The block producers on all blockchains have the power to select which transactions are included in blocks which gives them the ability to freeze accounts. EOS.IO software formalizes this authority by subjecting the process of freezing an account to a 17/21 vote of active producers. If the producers abuse the power they can be voted out and an account will be unfrozen.

Changing Account Code

When all else fails and an "unstoppable application" acts in an unpredictable manner, the EOS.IO software allows the block producers to replace the account's code without hard forking the entire blockchain. Similar to the process of freezing an account, this replacement of the code requires a 17/21 vote of elected block producers.

Constitution

The EOS.IO software enables blockchains to establish a peer-to-peer terms of service agreement or a binding contract among those users who sign it, referred to as a "constitution". The content of this constitution defines obligations among the users which cannot be entirely enforced by code and facilitates dispute resolution by establishing jurisdiction and choice of law along with other mutually accepted rules. Every transaction broadcast on the network must incorporate the hash of the constitution as part of the signature and thereby explicitly binds the signer to the contract.

The constitution also defines the human-readable intent of the source code protocol. This intent is used to identify the difference between a bug and a feature when errors occur and guides the community on what fixes are proper or improper.

Upgrading the Protocol & Constitution

The EOS.IO software define a process by which the protocol as defined by the canonical source code and its constitution, can be updated using the following process:

1. Block producers propose a change to the constitution and obtains 17/21 approval.
2. Block producers maintain 17/21 approval for 30 consecutive days.
3. All users are required to sign transactions using the hash of the new constitution.
4. Block producers adopt changes to the source code to reflect the change in the constitution and propose it to the blockchain using the hash of a git commit.
5. Block producers maintain 17/21 approval for 30 consecutive days.
6. Changes to the code take effect 7 days later, giving all full nodes 1 week to upgrade after ratification of the source code.
7. All nodes that do not upgrade to the new code shut down automatically.

By default configuration of the EOS.IO software, the process of updating the blockchain to add new features takes 2 to 3 months, while updates to fix non-critical bugs that do not require changes to the constitution can take 1 to 2 months.

Emergency Changes

The block producers may accelerate the process if a software change is required to fix a harmful bug or security exploit that is actively harming users. Generally speaking it could be against the constitution for accelerated updates to introduce new features or fix harmless bugs.

Scripts & Virtual Machines

The EOS.IO software will be first and foremost a platform for coordinating the delivery of authenticated messages to accounts. The details of scripting language and virtual machine are implementation specific details that are mostly independent from the design of the EOS.IO technology. Any language or virtual machine that is deterministic and properly sandboxed with sufficient performance can be integrated with the EOS.IO software API.

Schema Defined Messages

All messages sent between accounts are defined by a schema which is part of the blockchain consensus state. This schema allows seamless conversion between binary and JSON representation of the messages.

Schema Defined Database

Database state is also defined using a similar schema. This ensures that all data stored by all applications is in a format that can be interpreted as human readable JSON but stored and manipulated with the efficiency of binary.

Separating Authentication from Application

To maximize parallelization opportunities and minimize the computational debt associated with regenerating application state from the transaction log, EOS.IO separates validation logic into three sections:

1. Validating that a message is internally consistent;

2. Validating that all preconditions are valid; and
3. Modifying the application state.

Validating the internal consistency of a message is read-only and requires no access to blockchain state. This means that it can be performed with maximum parallelism. Validating preconditions, such as required balance, is read-only and therefore can also benefit from parallelism. Only modification of application state requires write access and must be processed sequentially for each application.

Authentication is the read-only process of verifying that a message can be applied. Application is actually doing the work. In real time both calculations are required to be performed, however once a transaction is included in the blockchain it is no longer necessary to perform the authentication operations.

Virtual Machine Independent Architecture

It is the intention of the EOS.IO software that multiple virtual machines can be supported and new virtual machines added over time as necessary. For this reason, this paper will not discuss the details of any particular language or virtual machine. That said, there are two virtual machines that are currently being evaluated for use within EOS.IO.

Web Assembly (WASM)

Web Assembly is an emerging web standard for building high performance web applications. With a few small modifications Web Assembly can be made deterministic and sandboxed. The benefit of Web Assembly is the widespread support from industry and that it enables contracts to be developed in familiar languages such as C or C++.

Ethereum developers have already begun modifying Web Assembly to provide suitable sandboxing and determinism in with their Ethereum flavored Web Assembly (WASM). This approach can be easily adapted and integrated with EOS.IO software.

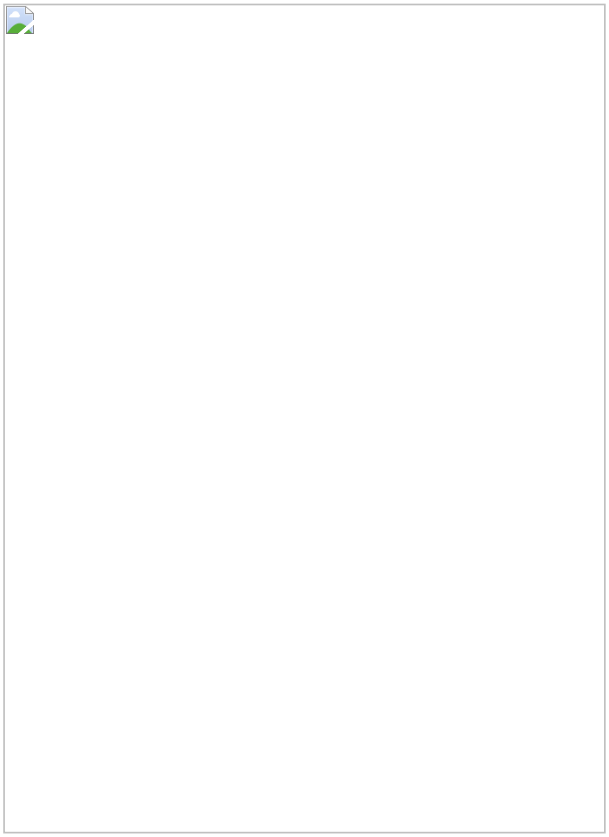
Ethereum Virtual Machine (EVM)

This virtual machine has been used for most existing smart contracts and could be adapted to work within an EOS.IO blockchain. It is conceivable that EVM contracts could be run within their own sandbox inside an EOS.IO blockchain and that with some adaptation EVM contracts could communicate with other EOS.IO blockchain applications.

Inter Blockchain Communication

EOS.IO software is designed to facilitate inter-blockchain communication. This is achieved by making it easy to generate proof of message existence and proof of message sequence. These proofs combined with an application architecture designed around message passing enables the details of inter-blockchain communication and proof validation to be hidden from application developers.

Merkle Proofs for Light Client Validation (LCV)



Integrating with other blockchains is much easier if clients do not need to process all transactions. After all, an exchange only cares about transfers in and out of the exchange and nothing more. It would also be ideal if the exchange chain could utilize lightweight merkle proofs of deposit rather than having to trust its own block producers entirely. At the very least a chain's block producers would like to maintain the smallest possible overhead when synchronizing with another blockchain.

The goal of LCV is to enable the generation of relatively light-weight proof of existence that can be validated by anyone tracking a relatively light-weight data set. In this case the objective is to prove that a particular transaction was included in a particular block and that the block is included in the verified history of a particular blockchain.

Bitcoin supports validation of transactions assuming all nodes have access to the full history of block headers which amounts to 4MB of block headers per year. At 10 transactions per second, a valid proof requires about 512 bytes. This works well for a blockchain with a 10 minute block interval, but is no longer "light" for blockchains with a 3 second block interval. The EOS.IO software enables lightweight proofs for anyone who has any irreversible block header after the point in which the transaction was included. Using the hash-linked structure shown below it is possible to prove the existence of any transaction with a proof less than 1024 bytes in size. If it is assumed that validating nodes are keeping up with all block headers in the past day (2 MB of data), then proving these transactions will only require proofs 200 bytes long.

There is little incremental overhead associated with producing blocks with the proper hash-linking to enable these proofs which means there is no reason not to generate blocks this way.

When it comes time to validate proofs on other chains there are a wide variety of time/ space/ bandwidth optimizations that can be made. Tracking all block headers (420 MB/year) will keep proof sizes small. Tracking only recent headers can offer a trade off between minimal long-term storage and proof size. Alternatively, a blockchain can use a lazy evaluation approach where it remembers intermediate hashes of past proofs. New proofs only have to include links to the known sparse tree. The exact approach used will necessarily depend upon the percentage of foreign blocks that include transactions referenced by merkle proof.

After a certain density of interconnectedness it becomes more efficient to simply have one chain contain the entire block history of another chain and eliminate the need for proofs all together. For performance reasons, it is ideal to minimize the frequency of inter-chain proofs.

Latency of Interchain Communication

When communicating with another outside blockchain, block producers must wait until there is 100% certainty that a transaction has been irreversibly confirmed by the other blockchain before accepting it as a valid input. Using EOS.IO software and DPOS with 3 second blocks and 21 producers, this takes approximately 45 seconds. If a chain's block producers do not wait for irreversibility it would be like an exchange accepting a deposit that was later reversed and could impact the validity of the a chain's consensus.

Proof of Completeness

When using merkle proofs from outside blockchains, there is a significant difference between knowing that all transactions processed are valid and knowing that no transactions have been skipped or omitted. While it is impossible to prove that all of the most recent transactions are known, it is possible to prove that there have been no gaps in the transaction history. The EOS.IO software facilitates this by assigning a sequence number to every message delivered to every account. A user can use these sequence numbers to prove that all messages intended for a particular account have been processed and that they were processed in order.

Conclusion

The EOS.IO software is designed from experience with proven concepts and best practices, and represents fundamental advancements in blockchain technology. The software is part of a holistic blueprint for a globally scalable blockchain society in which decentralised applications can be easily deployed and governed.